STRTRNS

Vulnerable to buffer overflows

Sean Barnum, Cigital, Inc. [vita[1]]

Copyright © 2007 Cigital, Inc.

2007-04-23

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 4650 bytes

Attack Category	• Malicious Input
Vulnerability Category	• Buffer Overflow • Unconditional
Software Context	• String Parsing
Location	• libgen.h
Description	The strtrns function will take the currentString and replace every instance of oldsegment with newsegment. The constructed string will be placed in newString. This function is a security risk because it is possible to overflow the newString buffer. If the currentString buffer is larger than the newString buffer, then an overflow will occur. Flag all instances of strtrns() as a potential vulnerability. Identify bounds checks for the function.

APIs	Function Name	Comments
	strtrns	

Method of Attack	strtrns() behaves like strcpy(), copying a source buffer into a destination buffer with certain translations. No bounds checking is done, so it is vulnerable to buffer overflow.
Exception Criteria	

Solutions	Solution Applicability	Solution Description	Solution Efficacy
	Always	Insert buffer overflow detection code both before and after the insertion and if condition is detected,	Effective.

---

1.   http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

		terminate when static sized buffers are used.	
	When using strtrns()	Use malloc for buffer creation, if possible.	Effective.

Signature Details	char * strtrns(const char *string, const char *old, const char *new, char *result);

Examples of Incorrect Code	```char currentString[30];
	char newString[20];
	// assignments
	strtrns(currentString, oldstuff,
	newstuff, newString);
	
	/*In this case, there is not
	enough space in newString for the
	data that is to be passed into
	it.*/```

Examples of Corrected Code	```/* the use of malloc in this case,
	would generally keep the buffer
	overflow condition from occurring
	*/
	/* malloc however may not always
	be a solution artifact available
	to the developer */
	
	#include <libgen.h>
	void main(int argc, char **argv)
	{
	char lower[] =
	"abcdefghijklmnopqrstuvwxyz";
	char upper[] =
	"ABCDEFGHIJKLMNOPQRSTUVWXYZ";
	char *buf;
	if(argc < 2) {
	printf("USAGE: %s arg\n",
	argv[0]);
	exit(0);
	} buf = (char
	*)malloc(strlen(argv[1]));
	strtrns(argv[1], lower, upper,
	buf);
	printf("%s\n", buf);
	}```

Source References	• Viega, John & McGraw, Gary. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X
Recommended Resources	

Discriminant Set	Operating System	• UNIX
	Languages	• C
		• C++

Cigital, Inc. Copyright

Copyright © Cigital, Inc. 2005-2007. Cigital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Cigital, including information about "Fair Use," contact Cigital at copyright@cigital.com[1].

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

---

1. mailto:copyright@cigital.com